Tampereen yliopisto

Kirsi Pietilä

ANKI COZMO -OHJELMOINTI

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintyö
Toukokuu 2019

TIIVISTELMÄ

Kirsi Pietilä: Anki Cozmo -ohjelmointi
Kandidaatintyö
Tampereen yliopisto
Tieto- ja sähkötekniikan TkK tutkinto-ohjelma
Toukokuu 2019

Cozmo on Ankin kehittämä sosiaalinen robotti, joka on suunniteltu lapsille leikkikaveriksi, mutta myös opetuskäyttöön ohjelmointimahdollisuuksiensa takia. Tässä työssä kerrotaan Cozmosta yleisesti, mutta pääpaino on Cozmoon liittyvässä ohjelmoinnissa. Ohjelmointikielenä toimii Python. Tässä työssä toteutetun ohjelman tarkoitus on saada Cozmo seuraamaan kuutiotaan.

Toteutetun ohjelman avulla Cozmo yrittää löytää kuution ja tunnistaa sen, jos se on tarpeeksi lähellä Cozmon näköpiirissä. Ohjelma laskee Cozmon ja kuution välisen etäisyyden ja kulman, joiden avulla määritellään Cozmon liikkeet kuutiota kohti. Cozmo reagoi muuttamalla suuntaansa, kun kuutio liikkuu sen näköpiirissä. Jos kuutio katoaa Cozmon näköpiiristä, se lähtee etsimään kuutiota uudestaan.

Toteutettu ohjelma suorittaa, mitä siltä vaadittiin. Ideaalinen ja reaaliaikainen se ei ole, sillä esimerkiksi Cozmo pysähtyy hetkeksi, jos kuutio liikkuu sen näköpiirissä ennen kuin se muuttaa suuntaansa. Joskus Cozmo kääntyy väärään suuntaan, eikä se aina tunnista kuutiota.

Avainsanat: Cozmo, robotiikkaparadigma

SISÄLLYSLUETTELO

1 JOHDANTO

Sosiaaliset robotit ovat olleet ihmisen mielikuvituksessa jo vuosikymmeniä, kuten ihmistä palvelemaan tehty Star Warsin C-3PO. C-3PO:n ensi esiintymisestä on yli 40 vuotta, ja sinä aikana ovat robotit tulleet entistä lähemmäksi ihmisiä. Teollisuuteen käytettävien robottien lisäksi markkinoille on tullut sosiaalisia robotteja, mutta vielä ne eivät toimi massatuotantonta kuluttajille.

Tässä työssä perehdytään lapsille ja opiskeluun suunniteltuun Cozmo-robottiin, joka on kuvassa 2.1. Cozmo on persoonalliseksi ohjelmoitu robotti, joka on suunniteltu olemaan vuorovaikutuksessa ihmisen kanssa, kuten pelaamaan pelejä. Cozmoa pystyy ohjelmoimaan. Työn tarkoituksena on antaa peruskattava kuvaus Cozmosta. Cozmoon perehdytään tavalla, jolla työntekijä tutustui siihen: kehittämällä Pythonilla toteutetun ohjelman. Työ sisältää siis paljon asiaa, joita työntekijä on huomioinut toimiessaan Cozmon kanssa. Ohjelman tarkoitus on saada Cozmo etsimään ja seuraamaan leikkikuutiotaan.

Teoriaosuudessa luvussa 2 käydään läpi tarkemmin Cozmo: mikä se on ja minkälainen se on. Sama luku käsittelee Cozmon lisäksi myös robotiikkaparadigmaa ja kaavoja, jotka ovat oleellisia totetutetun ohjelman kannalta. Luku 3 on omistettu kokonaan toteutetulle ohjelmalle, ja siitä käydään läpi kaikki tärkeimmät tiedot lukijan kannalta. Tuloksia käsitellään luvussa 4, miten käytännössä toteutettu ohjelma toimii ja kuinka ohjelmaa olisi voinut mahdollisesti kehittää.

2 TEORIA

Cozmo on robottien ja tekoälyn kehittämiseen erikoistunut yhtiö Ankin tuottama tuote. Pääsääntöisesti Cozmoa markkinoidaan lapsille leikkikaveriksi: sitä täytyy ruokkia ja leikkiä sen kanssa [9]. Cozmolla on 3 kuutiota, joiden avulla mahdollistetaan esimerkiksi Cozmon ja ihmisen välinen peli. Cozmo on ohjelmoitu Pythonilla, ja Anki antaa mahdollisuuden kokeilla ohjelmointia robottiin. Ohjelmoinnin voi tehdä Pythonilla tai aloittelijaystävällisemmällä graafisella Cozmo Code Labilla [4]. Jotta Cozmon kanssa voi toimia, tarvitaan älypuhelin, jossa on Cozmo-sovellus. Älypuhelimen kautta esimerksisi Cozmo kommunikoi, Cozmoa pyydetään peliin ja syötetään Python-ohjelma.

Vaikka Cozmo on tarkoitettu kuluttajille viihdykkeeksi, se on robotti. Tässä luvussa käsitellään robotiikkaparadigmaa, jonka avulla voidaan tarkkailla Cozmon käyttäytymistä. Viimeisessä aliluvussa esitetään matemaattiset kaavat, joiden avulla on toteutettu luvussa 3 esitetty ohjelma.

2.1 Anki Cozmo

Cozmo on kämmeneen sopiva robotti. Cozmon ketterä liikkuminen on mahdollistettu telamatoilla, jotka kiertävät Cozmon neljä rengasta. Huomattava fyysinen piirre Cozmossa on sen nostoteline, jota tässä työssä kutsutaan kauhaksi. Kauha on Cozmon "kädet", sen avulla se esimerkiksi nostaa ja kuljettaa kuutiota, mutta myös ilmaisee sosiaalisesti itseään. Toinen tapa, jolla Cozmo ilmaisee itseään, on sen päässä sijaitsevalla ruudulla kaksi nelikulmiota. Nelikulmiot toimivat Cozmon "silminä". Oikeina silminä toimii sen päässä sijaitseva 30 fps VGA -kamera, jonka näkymää käyttäjäkin pystyy tarkkailemaan. Kameraa on hyödynnetty niin, että Cozmolle on kehitetty ohjelmisto, joka mahdollistaa Cozmon tunnistamaan esimerkiksi ihmiset kasvoista. Päässä sijaitseva kaiutin mahdollistaa Cozmon puhumisen, ja tutut kasvot tunnistaessaan Cozmo osaa yhdistää ne annettuun nimeen ja osaa sanoa sen. Cozmon alapuolella sijaitsevat sensorit, jotka pystyvät tunnistamaan esimerkiksi pöydän reunan. Tällä vältetään Cozmon putoamista maahan. Cozmo ja Cozmon leikkikuutiot näkyvät kuvassa 2.1. [12]

Kuva 2.1. *Cozmo ja kuutiot.*

Cozmo on kuin elektroninen lemmikki, josta käyttäjän täytyy pitää huolta. Se ei myöskään toimi ilman Cozmo-sovellusta, jonka voi asentaa älypuhelimeen. Sieltä löytyvät käyttäjälle Cozmon "ruokinta"ja valmispelit, joita voi pelata Cozmon kanssa. Näitä toimintoja on tehtävä pitääkseen Cozmon tyytyväisenä, jos haluaa käyttää sitä vain "lemmikkinä". Näitä toimintoja voi välttää, jos on kiinnostunut vain käyttämään Cozmoa ohjelmoinnin kannalta. Ilman sovellusta ei voi syöttää yhtään ohjelmaa Cozmolle toteutettavaksi. Voidaankin siis sanoa älypuhelimen toimivan Cozmon älyllisenä keskuksena. Toimiva sovellus vaatii Cozmon oman Wi-Fi-yhteyden.

Cozmossa ei ole vaihdettavia pattereita, vaan se ladataan Cozmon omassa laturimessa, joka näkyy kuvassa 2.2. Cozmo pystyy tunnistamaan laturimen visuaalisesti, sillä siinä on identifioiva merkki. Cozmo pystyy myös tietämään, milloin se on siinä.

Cozmo voidaan luokitella sosiaaliseksi robotiksi, sillä se pystyy olemaan vuorovaikutuksessa käyttäjän kanssa. Cozmon kanssa kykenee esimerkiksi reaaliajassa pelaamaan pelejä Cozmo-sovelluksesta, ja sillä on paljon sosiaalisia reaktio-ominaisuuksia. Esimerkiksi, jos käyttäjä ei kopauta nyrkillään Cozmon kauhaa sen ollessa ylhäällä, Cozmo osoittaa surumielistä käyttäytymistä. Cozmoa voi ohjelmoida inhimilliseksi, kuten muokkaamalla sen kasvojen ilmettä tai sanomaan jotain.

Kuva 2.2. *Cozmon laturi.*

Cozmon koordinaatisto voidaan suhteuttaa suorakulmaiseen koordinaatistoon. Cozmon origo sijaitsee maassa kahden eturenkaan välissä. X-akseli suuntautuu suoraan Cozmon eteen, y-akseli on Cozmon vasen puoli ja z-akseli on Cozmosta ylöspäin. Cozmo käytännössä pystyy liikkumaan ja toimimaan vain xy-tasolla, koska se ei esimerkiksi pysty hyppimään muuttaakseen z-arvoa. Voidaan ajatella, että Cozmo pyörii z-akselin ympäri, joten sen avulla voidaan määrittää Cozmon kierto xy-tasolla. [3]

Cozmo SDK (engl. software development kit) on robotiikkaan perustuva alusta, jota käytetään Cozmon ohjelmointiin [11]. Cozmo SDK tarjoaa kuluttajille ohjelmointirajapinnan, jossa määritellään kaikki luokat, funktiot ja komennot, joita voi hyödyntää Cozmon kanssa. Luvussa 3 esitetään tarkemmin ohjelmassa hyödynnettyjä funktioita.

2.1.1 Cozmon leikkikuutiot

Cozmolla on käytettävissä 3 leikkikuutiota, jotka ovat tärkeä osa Cozmon elämää. Kuutioilla on tahkoissa identifioiva kuvio, kuten kuvasta 2.1 voi nähdä. Jokaisessa kuutiossa on 4 lediä, joita käyttäjä voi kontrolloida [1]. Tärkeä ominaisuus kuutioissa on sensorit, joiden avulla pystyy määrittämättään, liikkuuko kuutio tai onko kuutiota kopautettu. Cozmo pystyy tunnistamaan kuutiot kahdella tavalla:

1. Visuaalisesti: Cozmo tunnistaa kuutiot toisistaan tahkokuvien avulla.
2. Langattomasti: Sensorien avulla Cozmo pystyy muodostamaan yhteyden kuutioihin, jos ne ovat tarpeeksi lähellä.

Cozmon visuaalinen tunnistaminen on myös rajallinen. Se pystyy tunnistamaan kuution noin 0,5 m asti. Kuutiot toimivat pattereilla, ja langaton tunnistaminen vaatii patterin.

2.1.2 Cozmon käyttöönotto Python-ohjelmia varten

Cozmo vaatii tietokoneen ja älypuhelimen, jos haluaa syöttää Cozmolle Python-ohjelmia. Älypuhelin täytyy olla kytköksissä tietokoneeseen USB-johdolla, ja Cozmo yhdistetään puhelimeen Wi-Fi-yhteyden kautta [6]. Cozmo SDK:n käyttöönotto riippuu käyttäjän tietokoneen ja puhelimen käyttöjärjestelmistä [7]. Asennukset tapahtuvat eri tavalla esimerkiksi Windowsille ja Linuxille tai Androidille ja iOS:lle.

Python-ohjelmia varten täytyy älypuhelimeen asentaa Cozmo-sovellus. Yhteyden luomiseksi täytyy sovellus käynnistää, ja sen asetuksista painaa *Enable SDK* -nappia. Komentorivin käyttäminen on tärkeä osa on laitteiden käyttöjärjestelmistä riippumatta. Komentorivin avulla Cozmolle pystyy syöttämään ohjelmia, joko valmiita tai omia. Olennaista on olla kansiossa, jossa haluttu ohjelma sijaitsee. Ohjelma on hyvä olla toteutettu Python 3.5.1 -versiolla tai myöhemmällä [7].

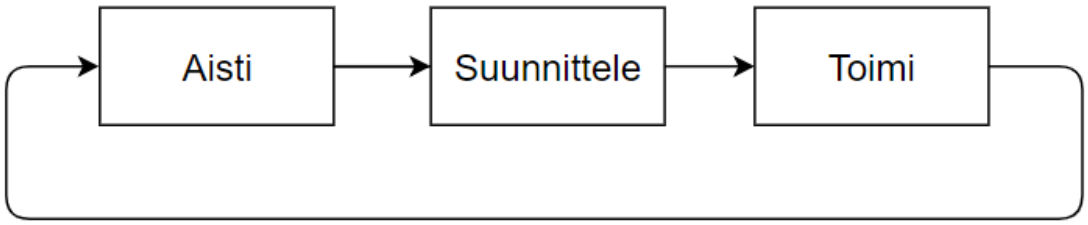
2.2 Robotiikkaparadigma

Robotiikka-paradigma (engl. robotic paradigm) on ongelman ratkaisutapa, jolla määritellään robotin toimintaa ja älykkyyttä. Robottien primitiiviset toiminnat voidaan jakaa kolmeen kategoriaan, joihin robotin määrittelevät funktiot voidaan luokitella:

1. Aisti (engl. sense): Jos funktio saa informaatiota robotin sensoreista, ja informaatio on hyödyllinen muille funktioille.
2. Suunnittele (engl. plan): Jos funktio ottaa vastaan informaatiota ja tuottaa sen avulla tehtäviä robotille.
3. Toimi (engl. act): Toteuttaa tehtävän fyysisessä maailmassa.

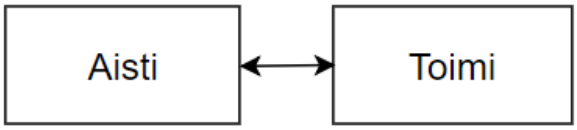
Edellä mainittujen kategorioiden avulla voidaan määritellä kolme paradigmaa: hierarkkinen (engl. hierarchical), reaktiivinen (engl. reactive) ja hybridi pohtiva/reaktiivinen (engl. hybrid deliberative/reactive). [10, s. 5]

Hierarkkinen paradigma on vanhin metodi robotiikan historiassa. Siinä käytetään kaikkia kolmea kategoriaa, ja toiminnalla on vain yksi suunta, kuten kuvasta 2.3 voidaan nähdä. Ensin robotti aistii ympäristöään, ja jos se saa tarvitsemaansa informaatiota, se suunnittelee toimintansa ja lopulta toteuttaa suunnittelun. [10, s. 6]



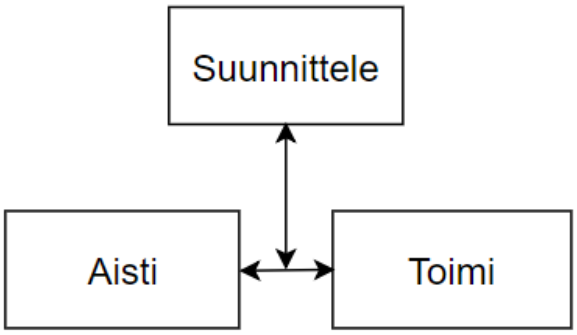
Kuva 2.3. *Hierarkkisen paradigman toimintamalli.*

Reaktiivisessa paradigmassa on luovuttu suunnittelusta ja se koostuu vain aisti-toimi -parista. Toimintamalli on esitetty kuvassa 2.4. Suunnittelusta luopuminen tuottaa huomattavasti tästä paradigmasta nopeamman kuin hierarkkisesta. Tässä paradigmassa robotilla on useita aisti-toimi -pareja. Nämä parit voivat toimia samanaikaisesti, mutta robotti yleensä toimii parien yhdistelmällä. Esimerkiksi eräs aisti-toimi -pari voi olla "aja eteenpäin kohti kohdetta"ja toinen "väistä estettä kääntymällä 90°", joiden yhdistelemällä robotti kääntyy väliaikaisesti 45° kurssiltaan. [10, s. 8–9]



Kuva 2.4. *Reaktiivisen paradigman toimintamalli.*

Reaktiivinen paradigma toimii pohjana hybridi-mallille, mutta siinä otetaan huomioon jälleen suunnittelu, kuten kuvassa 2.5 on esitetty. Aluksi robotti suunnittelee, kuinka hajauttaa tehtävän alitehtäviin ja mitkä ovat parhaimmat toiminnot alitehtävien saavuttamiseksi.



Kuva 2.5. *Hybridi paradigman toimintamalli.*

Suunnittelun jälkeen robotti käyttäytyy kuten reaktiivisessa paradigmassa. [10, s. 9–10]

2.3 Matemaattiset kaavat ohjelmaan

Suunniteltussa ohjelmassa, jossa Cozmo seuraa kuutiotaan, on hyväksikäytetty kahta vektoria, joiden avulla määritellään minne Cozmon pitää mennä. Tässä aliluvussa käsitellään tarkemmin matemaattisia kaavoja, joita on hyödynnetty Cozmon suunnistuksessa. Luvussa 3 käsitellään tarkemmin näiden kahden vektorin merkitystä ohjelmassa. Ohjelmassa käytettävä matematiikka on suhteellisen suoraviivaista, mutta erittäin olennainen osa ohjelman toimivuuden kannalta.

Olkoon vektorit $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

Vektorin normi tai pituus määritellään seuraavasti

$$\|\mathbf{a}\| = \sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}. \tag{2.1}$$

Kahden vektorin $\mathbf{a}$ ja $\mathbf{b}$ pistetulo saadaan

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n. \tag{2.2}$$

Vektorien $\mathbf{a}$ ja $\mathbf{b}$ välinen kulma $0 \leq \theta \leq \pi$ määritellään

$$cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \, \|\mathbf{b}\|}. \tag{2.3}$$

[8] Olkoon vektorit $\mathbf{v}, \mathbf{u} \in \mathbb{R}^2$, ja niistä muodostuva matriisi

$$A = \begin{bmatrix} v_1 & v_2 \\ u_1 & u_2 \end{bmatrix} \tag{2.4}$$

Matriisin A determinantti saadaan

$$det(A) = \begin{vmatrix} v_1 & v_2 \\ u_1 & u_2 \end{vmatrix} = v_1 u_2 - v_2 u_1. \tag{2.5}$$

Determinantin avulla voidaan tarkastella vektorien sijoittumista toistensa suhteen. Jos

determinantti on negatiivinen, vektori $\mathbf{u}$ on vektorin $\mathbf{v}$ oikealla puolella. Positiivinen determinantti tarkoittaa taas vektorin $\mathbf{u}$ olevan vektorin $\mathbf{v}$ vasemmalla puolella. [5]

3 KOHTEEN SEURAAMINEN

Toteutettu ohjelma on tehty kahdesta syystä: tapa tutustua Cozmoon ja saada Cozmo seuraamaan leikkikuutiota. Tarkoituksena on toteuttaa suhteellisen reaaliajassa toimiva seuraaminen. Ohjelma toteutetaan hyödyntäen Cozmon valmista rajapintaa ja matemaattisia kaavoja. Rajapintaa ja sieltä hyödynnettyjä asioita käydään tässä luvussa tarkemmin lävitse, mutta matemaattiset kaavat on esitetty jo luvussa 2. Ohjelma on toteutettu Pythonilla ja se on nähtävissä liitteessä A.

3.1 Ohjelman yleiskuvaus

Aivan aluksi Cozmo lähtee etsimään kuutiota. Se kiertää 360° itsensä ympäri, ja yrittää aktiivisesti löytää kuution. Jos Cozmo näkee kameransa avulla kuution ja pystyy tunnistamaan sen, Cozmo lähtee ajamaan sitä kohti. Ohjelma laskee kuution ja robotin välisen etäisyyden ja kulman, minne Cozmon täytyy kääntyä.

Cozmo reagoi, jos kuutio liikkuu sen näköpiirissä tai katoaa sen näköpiiristä. Molemmissa tapauksissa se pysähtyy. Ensimmäisessä tapauksessa Cozmo kääntyy näköpiirissä olevaa kuutiota kohti ja lähtee ajamaan sitä kohti. Viimeisessa tapauksessa Cozmo lähtee etsimään uudestaan kadonnutta kuutiota. Tätä jatkuu niin kauan, kunnes käyttäjä lopettaa ohjelman komentoriviltä.

Cozmo reagoi sosiaalisesti, jos se ei löydä kuutioita kierrettyään itsensä ympäri tai kun Cozmo löytää sen. Ensimmäisessä tapauksessa Cozmo suuttuu "puhumalla"kiukkuisesti ja heiluttelemalla kauhaansa. Toisessa tapauksessa Cozmo osoittaa tyytyväisyyttä laajentamalla silmiään.

Ohjelma perustuu käytännössä ikuiseen silmukkaan, jossa tarkkaillaan parametria cube_not_lost. Jos tämä on epätosi, Cozmo ei visuaalisesti pysty tunnistamaan kuutiota ja lähtee etsimään sitä. Jos taas se on tosi, ohjelma laskee Cozmolle kulman ja etäisyyden, jonka ajaa.

3.2 Cozmo.event-rajapinta

Cozmo SDK toimii paljon tapahtumien (engl. event) ympärillä: niiden lähettämiseen ja tarkkailuun. Cozmo.event toteuttaa tapahtumien lähetyksen ja auttaa niiden käsittelyssä.

Tapahtumat syntyvät useimmiten, kun Cozmon tunnistamille objekteille tehdään jotain. Objekteja voivat olla esimerkiksi kuutio tai käyttäjän määrittelemä esine. Tässä työssä käytettyjä tapahtumia käsitellään myöhemmin.

Tapahtumia voidaan käsitellä eri tavalla, mutta tässä työssä käytetään add_event_handler funktiota. Sen avulla tapahtumat huomioidaan aina, kun jokin laukaisee ne. Tapahtuman lisäksi add_event_handler ottaa toiseksi parametrikseen kutsuttavan funktion. Tämä käytännössä tarkoittaa sitä, että kun tapahtuma on aiheutettu, siirrytään kutsuttavaan funktioon. [2]

3.3 Tapahtumat

Rajapinta cozmo.objects määrittelee luokat ja tapahtumat, joiden avulla mahdollistetaan kuution seuraaminen. Tämän rajapinnan avulla Cozmo pystyy tunnistamaan kuutionsa, mutta myös muita objekteja esimerkiksi käyttäjän määrittämä asia. Objektit ja kuutiot voivat lähettää tietynlaisia tapahtumia, jotka laukeavat asioiden tapahduttua. Esimerkiksi tässä työssä on 3 oleellista tapahtumaa, joiden avulla mahdollistetaan Cozmon reagoiminen: cozmo.objects.EvtObjectAppeared, cozmo.objects.EvtObjectDisappeared ja cozmo.objects.EvtObjectMoving. Nämä tapahtumat tulevat esille aikaisemmin esitetyn add_event_handler avulla, joka ohjaa oikeisiin funktioihin tekemään haluttuja asioita. [1]

Cozmo.objects.EvtObjectAppeared aiheuttaa tapahtuman, kun Cozmo pystyy tunnistamaan visuaalisesti objektin. Tunnistuksen avulla pystytään määrittämään kohteen sijainti [1]. Kun tämä tapahtuma aiheutuu, siirrytään ohjelmassa määriteltyyn funktioon handle_object_appeared. Funktiossa identifioidaan Cozmon näkemä kuutio ja asetetaan se globaaliksi muuttujaksi. Tunnistuksen myötä määritellään myös globaali muuttuja cube_not_lost todeksi.

Cozmo.objects.EvtObjectDisappeared-luokassa tapahtuu edeltävälle vastakkainen tapahtuma, kun Cozmo ei pysty enää aikaisemmin tunnistamaansa objektia visuaalisesti näkemään/tunnistamaan. Toisin kun edellisessä määriteltiin kuutio, tässä tehdään vastakkaisesti. Tapahtuman myötä siirrytään funktioon handle_object_disappeared, missä globaalisiti määritelty kuutio ei ole enää määritelty ja muuttuja cube_not_lost asetetaan epätodeksi. Funktiossa kutsutaan myös Cozmo SDK:n tarjoamaa toimintoa, joka lopettaa kaikki Cozmon tekemät toimenpiteet. Esimerkiksi, jos Cozmo ajaa kohti kuutiotaan, tämän myötä se pysähtyy ja lopettaa kyseisen toimenpiteen.

Cozmo.objects.EvtObjectMoving taas laukaisee tapahtuman, kun aktiivinen (tässä tapauksessa tunnistettu) objekti liikkuu [1]. Kuutioiden tapauksessa liikkuminen pystytään tunnistamaan niiden sisällä olevien kiihtyvyysanturien avulla. Tapahtuman aiheuttaa handle_object_moving_started funktiokutsun, missä tehdään asioita vain, jos kuutio on edelleen Cozmon näköpiirissä. Myös tässä funktiossa Cozmo lopettaa kaikki toimintonsa, ja asettaa kuution ja Cozmon välisen etäisyyden kertovan muuttujan tyhjäksi.

3.4 Cozmon navigointi

Cozmon ja kuution sijaintitiedot on mahdollista saada käytettäväksi. Ohjelma määrittelee Cozmon ja kuution välisen vektorin, kun Cozmo pystyy tunnistamaan visuaalisesti kuution. Kuutioon ohjaava vektori muodostetaan vähentämällä Cozmon sijaintipiste kuution sijaintipisteestä. Ohjelma määrittelee myös Cozmolle oman vektorin, jonka avulla voidaan saada selville mihin suuntaan Cozmon kasvot osoittavat.

Näiden kahden vektorin avulla voidaan laskea niiden välinen kulma kaavan (2.3) mukaisesti. Tämän avulla tiedetään kulman suuruus, minkä Cozmo täytyy kääntyä, jotta sen kasvot osoittavat suoraan kuutiota kohti. Kaava (2.3) ei riitä kertomaan oikeaa suuntaa, toisin sanoen, on mahdotonta tietää vain tämän perusteella onko kuutio alunperin Cozmon vasemmalla vai oikealla puolella. Ongelma ratkaistaan determinantin avulla kaavan (2.5) mukaisesti. Determinantti lasketaan robottivektorille ja kuutioon ohjaavalle vektorille, minkä avulla tiedetään kääntyykö Cozmo vasemman vai oikean kulman mukaan. Jos determinantti on negatiivinen, kuutio on Cozmon vasemmalla puolella, joten käännettävä kulma on positiivinen. Jos taas determinantti on positiivinen, kuutio on Cozmon oikealla puolella ja kulma on negatiivinen.

Etäisyys kuution ja Cozmon välillä saadaan kuutioon ohjaavaan vektorista kaavan (2.1) avulla. Etäisyys on matka, joka syötetään Cozmolle ajettavaksi, kun oikea kulma on selvitetty.

4 PÄÄTELMÄT

Toteutettu ohjelma ei toimi niin ideaalisesti kuin se on esitetty edellisessä luvussa 3. Ohjelma ei toimi reaaliajassa, sillä Cozmon toiminta ei ole aina suoraviivaista. Esimerkiksi funktioissa handle_object_disappeared ja handle_object_moving_started pakotetaan Cozmo lopettamaan kaikki toimintonsa, joka aiheuttaa pientä viivettä ennen kuin Cozmo jatkaa toimintojaan.

Toisinaan Cozmo ei tunnista kuutiota, vaikka se olisi alle puolen metrin etäisyydellä. Joskus syy johtuu siitä, että kuutio on liian lähellä Cozmon kameraa, mutta aina kyse ei ole siitä. Ajottain Cozmo häiriintyy esimerkiksi, jos se tunnistaa laturimen.

Eniten päävaivaa tuottava asia on, jos Cozmo kääntyy väärään suuntaan. Lähtökohtaisesti aina, kun Cozmo löytää kuution, se kääntyy oikeaan suuntaan. Kun ensimmäisen kerran liikauttaa kuutiota Cozmon näköpiirissä, Cozmo pysähtyy, mutta ei käänny. Toisen kuution liikautuksen jälkeen Cozmo yleensä kääntyy, mutta väärään suuntaan. Tämän jälkeen Cozmo kääntyy lähes poikkeuksetta vastakkaiseen suuntaan mitä pitäisi.

Toisinaan taas Cozmo tekee sen, mitä ohjelma vaatii. Asia voi riippua käyttäjästä, kuinka esimerkiksi heiluttelee kuutioita Cozmon näköpiirissä. Cozmo tunnistaa parhaiten kuution, kun se on mahdollisimman keskellä sen näköpiiriä. Se ei esimerkiksi tunnista puoliksi näkyvää kuutiota.

Luvussa 2 esiteltiin robotiikka-paradigmasta ja sen kolmesta eri toimintamallista roboteille. Aistiin voidaan luokitella Cozmon tunnistaessa kuutio, kuutio liikkuu tai kuutio katoaa Cozmon näköpiiristä. Tapahtumien toimeentullessa, Cozmo reagoi niihin määrättyjen asioiden mukaisesti omissa funktioissaan. Tämän perusteella voitaisiin muodostaa muutamia aisti-toimi-pareja reaktiivisen paradigman mukaisesti, mutta ero on siinä, että Cozmo ei toimi näiden yhdistelminä, jos kaksi aisti-toimi-paria toimivat samanaikaisesti. Tässä toteutetut tapahtumat ovat hyvin irrallisia toisistaan siinä mielessä, että niillä on vain selvä tarkoitus mitä tehdä. Jos ohjelmaa muuttaisi ideaalisemmaksi, esimerkiksi Cozmo ei lopettaisi kaikkia toimintojaan kuution liikkuessa, voisi reaktiivista paradigmaa ehkä hyödyntää paremmin.

Cozmo kuuluu tämän ohjelman puitteissa hierarkkiseen paradigmaan, koska sen toiminta on hyvin suoraviivaista. Kaikki aistimus liittyy kuution paikantamiseen ja sen informaation avulla Cozmo "suunnittelee"mitä se tekee ja lopulta toteuttaa suunnitelman.

Tätä ohjelmaa voisi luultavasti parantaa käyttämällä tarkempia paikannus- ja tunnistus-

menetelmiä sillä, kun kuutio liikkuu, tapahtuma ei aina aiheudu. Tässä työssä kuution liikkuminen tarkasteltiin kuution sisällä olevan anturin avulla. Laskennalliset menetelmät voivat viivästyttää Cozmon toimintaa, ja tämä robotti on rajallinen toimintojensa kanssa. Sillä esimerkiksi, jos Cozmolle asettaa paljon erilaisia toimintoja, kuten puhumista, sen käyttäytyminen hidastuu huomattavasti.

LÄHDELUETTELO

[1] *Cozmo SDK. cozmo.objects*. Anki. URL: http://cozmosdk.anki.com/docs/generated/cozmo.objects.html#module-cozmo.objects/ (viitattu 03. 04. 2019).

[2] *Cozmo SDK. cozmo.event*. Anki. URL: http://cozmosdk.anki.com/docs/generated/cozmo.event.html#/ (viitattu 03. 04. 2019).

[3] *Cozmo Util*. Anki. URL: http://cozmosdk.anki.com/docs/generated/cozmo.util.html#module-cozmo.util (viitattu 03. 04. 2019).

[4] *Create with Cozmo. Cozmo Code Lab*. Anki. URL: https://www.anki.com/en-us/cozmo/create-with-cozmo/ (viitattu 20. 03. 2019).

[5] *Dot Product, Cross Product, Determinants*. University of Maryland. College Park. URL: http://www2.math.umd.edu/~petersd/241/crossprod.pdf.

[6] *Getting Started with the SDK*. Anki. URL: https://developer.anki.com/blog/learn/tutorial/getting-started-with-the-cozmo-sdk/ (viitattu 01. 04. 2019).

[7] *Initial Setup*. Anki. URL: http://cozmosdk.anki.com/docs/initial.html (viitattu 03. 04. 2019).

[8] T. Kaarakka. *Matriisilaskentaa insinöörien tarpeisiin*. Saatavissa: https://moodle2.tut.fi/ > MAT-01200 > Yleinen > Luentomoniste. Tampereen teknillinen yliopisto. Tampere, 2018.

[9] *Life with Cozmo*. Anki. URL: https://anki.com/en-us/cozmo/life-with-cozmo/nurture.html (viitattu 11. 05. 2019).

[10] R. R. Murphy. *Introduction to AI robotics*. English. Cambridge (MA): MIT Press, 2000. (Viitattu 04. 04. 2019).

[11] *SDK for Anki Cozmo*. Anki. URL: https://pypi.org/project/cozmo/ (viitattu 22. 03. 2019).

[12] *Story of Cozmo. Cozmo's tech*. Anki. URL: https://www.anki.com/en-gb/cozmo/cozmo-technology/ (viitattu 03. 04. 2019).

A OHJELMA

Liittenä on työssä toteutettu ohjelmakoodi, jossa Cozmo etsii ja seuraa kuutiota.

```
1  def cozmo_program(cozmo_robot: cozmo.robot.Robot):
2
3      global robot
4      global cube
5
6      init_robot(cozmo_robot)
7
8      robot.set_head_angle(degrees(0)).wait_for_completed()
9
10     robot.add_event_handler(cozmo.objects.EvtObjectAppeared,
11                             handle_object_appeared)
12     robot.add_event_handler(cozmo.objects.EvtObjectDisappeared,
13                             handle_object_disappeared)
14     robot.add_event_handler(cozmo.objects.EvtObjectMovingStarted,
15                             handle_object_moving_started)
16
17     while True:
18
19         if not cube_not_lost:
20             find_cube()
21
22         else:
23             distance_to_cube = turn_to_cube()
24             init_distance(distance_to_cube)
25             drive_to_cube()
26
27         time.sleep(.1)
28
29 cozmo.run_program(cozmo_program, use_viewer=True,
30                             force_viewer_on_top=True)
```

Funktio A.1. *Ohjelman mainfunktio.*

```
1  def find_cube():
2
3      global cube
4
5      degrees_to_turn = 20
6      counter = 0
7      max_turns = 360/degrees_to_turn
8
9      while cube is None:
10         robot.turn_in_place(degrees(degrees_to_turn))\
11             .wait_for_completed()
12
13         # Cozmo has turned 360 degrees and didn't find a cube.
14         if counter == max_turns:
15             break
16
17         counter += 1
18
19     if cube is not None:
20         robot.play_anim_trigger(cozmo.anim.Triggers.BlockReact)\
21             .wait_for_completed()
22
23     else:
24         robot.play_anim_trigger(cozmo.anim.Triggers.MajorFail)\
25             .wait_for_completed()
```

Funktio A.2. *Kuutioon etsintään toteutettu funktio.*

```
1  def turn_to_cube():
2
3      head_to_cube = (cube.pose.position.x - robot.pose.position.x,
4                      cube.pose.position.y - robot.pose.position.y)
5      robot_vector = (cos(robot.pose.rotation.angle_z.radians),
6                      sin(robot.pose.rotation.angle_z.radians))
7
8      angle = radians(angle_to_turn(head_to_cube, robot_vector))
9      robot.turn_in_place(angle).wait_for_completed()
10
11     return distance_mm(norm(head_to_cube))
```

Funktio A.3. *Funktio, jonka avulla Cozmo kääntyy kuutiota kohti.*

```
1 def angle_to_turn(a, b):
2     # head_to_cube's and robot_vector's angle
3     angle = angle_between_vectors(a,b)
4     det = determinant(a,b)
5
6     if det < 0:
7         return angle
8     else:
9         return -angle
```

Funktio A.4. *Määrittelee oikean kulman.*

```
1 def angle_between_vectors(a, b):
2     cosx = dot_product(a,b)/(norm(a)*norm(b))
3     rad = acos(cosx)
4
5     return rad
```

Funktio A.5. *Laskee vektorien välisen kulman.*

```
1 def norm(a):
2     return sqrt(a[0]**2+a[1]**2)
3
4 def dot_product(a,b):
5     return a[0]*b[0]+a[1]*b[1]
6
7 def determinant(a,b):
8     return a[0]*b[1]-a[1]*b[0]
```

Funktio A.6. *Kaavoja, joita käytetty laskennassa.*

```
1 def drive_to_cube():
2     if cube_not_lost:
3         robot.drive_straight(distance, speed_mmps(40)).\
4             wait_for_completed()
```

Funktio A.7. *Funktio, jossa Cozmo määritetään ajamaan kuutiota kohti.*

```
1 def handle_object_appeared(evt, **kw):
2     cube_found = robot.world.get_light_cube(evt.obj.cube_id)
3     init_cube(cube_found)
4
5     global cube_not_lost
6     cube_not_lost = True
```

Funktio A.8. *Tapahtumafunktio, kun kuutio tulee Cozmon näköpiitiin.*

```
1  def handle_object_disappeared(evt, **kw):
2      global cube
3      global distance
4      global cube_not_lost
5      cube = None
6      distance = None
7      cube_not_lost = False
8
9      robot.abort_all_actions()
```

Funktio A.9. *Tapahtumafunktio, kun kuutio katooa Cozmon näköpiiristä.*

```
1  def handle_object_moving_started(evt, **kw):
2      if cube_not_lost:
3          robot.abort_all_actions()
4          global distance
5          distance = None
```

Funktio A.10. *Tapahtumafunktio, kun kuutio liikkuu.*

```
1  def init_robot(cozmo_robot: cozmo.robot.Robot):
2      global robot
3      robot = cozmo_robot
4
5  def init_cube(cube_found):
6      global cube
7      cube = cube_found
8
9  def init_distance(distance_to_cube):
10     global distance
11     distance = distance_to_cube
```

Funktio A.11. *Apufunktiota, kun määritellään globaaleja muuttujia.*